OIPE IAP17
SEP 29 2006
PATENT & TRADEMARK OFFICE

Docket No.: 03226/442001; P7527

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

32615
PATENT TRADEMARK OFFICE

In re Patent Application of:
Karine Excoffier

Application No.: 10/613,846

Confirmation No.: 7257

Filed: July 3, 2003

Art Unit: 2132

For: MULTIPLE PASSWORD POLICIES IN A DIRECTORY SERVER SYSTEM

Examiner: T. R. Peeso

SUBMISSION OF CERTIFIED COPY OF PRIORITY DOCUMENT

MS Issue Fee
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Concurrent with the filing of the above referenced patent application, Applicants claimed priority under 35 U.S.C. 119 to the following prior foreign application filed in the following foreign country on the date indicated below:

Country	Application No.	Date
France	0208489	July 5, 2002

This foreign priority claim was acknowledged by the U.S. Patent and Trademark Office in its official Filing Receipt dated October 3, 2003. In support of this claim, a **certified copy** of the said original foreign application is filed herewith.

BEST AVAILABLE COPY

173637

Applicants believe no fee is due with this response. However, if a fee is due, please charge our Deposit Account No. 50-0591, under Order No. 03226/442001 from which the undersigned is authorized to draw.

Dated: September 29, 2006

Respectfully submitted,

By [illegible]

for Robert P. Lord T. Chyau Liang
Registration No.: 46,479 48,885
OSHA · LIANG LLP
1221 McKinney St., Suite 2800
Houston, Texas 77010
(713) 228-8600
(713) 228-8778 (Fax)

02.08489
Letter (1)

R E P U B L I Q U E   F R A N Ç A I S E

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 30 MAI 2003

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

[illegible]

Martine PLANCHE

CERTIFIED COPY OF PRIORITY DOCUMENT

10/613,846

DB 267/141102

1er dépôt

INPI
INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE
26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 94 86 54

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI

cerfa
N° 11354*01

REQUÊTE EN DÉLIVRANCE 1/2

Important ! Remplir impérativement la 2ème page.

Cet imprimé est à remplir lisiblement à l'encre noire

DB 540 W / 190600

Réservé à l'INPI	
REMISE DES PIÈCES DATE	5 JUIL 2002
LIEU	75 INPI PARIS
N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI	0208489
DATE DE DÉPÔT ATTRIBUÉE PAR L'INPI	05 JUIL. 2002
Vos références pour ce dossier *(facultatif)*	SUN 37 (120709)

1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE

CABINET NETTER
36 avenue Hoche
75008 PARIS

Confirmation d'un dépôt par télécopie ☐ N° attribué par l'INPI à la télécopie

2 NATURE DE LA DEMANDE	**Cochez l'une des 4 cases suivantes**	
Demande de brevet	☒	
Demande de certificat d'utilité	☐	
Demande divisionnaire	☐	
Demande de brevet initiale	N°	Date / /
ou demande de certificat d'utilité initiale	N°	Date / /
Transformation d'une demande de brevet européen *Demande de brevet initiale*	☐ N°	Date / /

3 TITRE DE L'INVENTION (200 caractères ou espaces maximum)

Multiple password policies in a directory server system.

4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE	Pays ou organisation Date / /	N°
	Pays ou organisation Date / /	N°
	Pays ou organisation Date / /	N°
	☐ **S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»**	

5 DEMANDEUR	☐ **S'il y a d'autres demandeurs, cochez la case et utilisez l'imprimé «Suite»**	
Nom ou dénomination sociale	SUN MICROSYSTEMS, INC	
Prénoms		
Forme juridique		
N° SIREN		
Code APE-NAF		
Adresse — Rue	901 San Antonio Road	
Adresse — Code postal et ville	94303	PALO ALTO Californie
Pays	Etats-Unis d'Amérique	
Nationalité	Société des Etats-Unis d'Amérique	
N° de téléphone *(facultatif)*		
N° de télécopie *(facultatif)*		
Adresse électronique *(facultatif)*		

INPI
INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

CERTIFICAT D'UTILITÉ

REQUÊTE EN DÉLIVRANCE 2/2

Réservé à l'INPI

REMISE DES PIÈCES
DATE 5 JUIL 2002
LIEU 75 INPI PARIS
N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0208489

DB 540 W / 190600

Vos références pour ce dossier : *(facultatif)*	SUN 37 (120709)
6 MANDATAIRE	
Nom	PLAÇAIS
Prénom	Jean-Yves
Cabinet ou Société	Cabinet NETTER
N °de pouvoir permanent et/ou de lien contractuel	
Adresse — Rue	36 avenue Hoche
Adresse — Code postal et ville	75008 Paris
N° de téléphone *(facultatif)*	01 58 36 44 22
N° de télécopie *(facultatif)*	01 42 25 00 45
Adresse électronique *(facultatif)*	
7 INVENTEUR (S)	
Les inventeurs sont les demandeurs	☐ Oui [✓ Non] **Dans ce cas fournir une désignation d'inventeur(s) séparée**
8 RAPPORT DE RECHERCHE	**Uniquement pour une demande de brevet (y compris division et transformation)**
Établissement immédiat ou établissement différé	☐ immédiat [✓ différé]
Paiement échelonné de la redevance	**Paiement en deux versements, uniquement pour les personnes physiques** ☐ Oui ☐ Non
9 RÉDUCTION DU TAUX DES REDEVANCES	**Uniquement pour les personnes physiques** ☐ Requise pour la première fois pour cette invention *(joindre un avis de non-imposition)* ☐ Requise antérieurement à ce dépôt *(joindre une copie de la décision d'admission pour cette invention ou indiquer sa référence)* :
Si vous avez utilisé l'imprimé «Suite», indiquez le nombre de pages jointes	

10 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire) N° Conseil 92-1197 (B) (M) Jean-Yves PLAÇAIS [illegible]	**VISA DE LA PRÉFECTURE OU DE L'INPI** M MARTIN

La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire. Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

Multiple password policies in a directory server system

This invention relates to distributed computer systems.

In certain fields of technology, complete computer systems, including a diversity of equipments, are optimized for storing and retrieving data. Such systems may provide services to user machines related to a local network, e.g., an Intranet, or to a global network, e.g., the Web network.

It is desirable that network users can access, upon a query, to a large number of data, making it possible for the network users to create their own dynamic web site or to consult a dynamic web site, for example an e-commerce site on a multi platform computer system (Solaris, Windows NT). These queries are directed to a directory, e.g., a LDAP directory, and managed by a directory server. It is further desirable that this access to a large number of data be made possible more rapidly for each query arriving after a first query.

A general aim of the present invention is to provide advances in these directions.

Broadly, there is proposed a method of implementing a password checking function based on password-related data in a directory server system. The directory server system has a directory server interacting with entries organized in a tree structure. The entries comprising user entries. The method comprises the steps of :
a) creating an additional entry, having attached password-related data,
b) attaching extra data to a given user entry, the extra data designating the additional entry, and
c) upon a call of the password checking function for the given entry, executing the password policy checking function for the given user entry based on the password-related data in the additional entry designated by the extra data of the given user entry.

There is also proposed a directory server capable of interacting with entries organized in a tree structure in a directory server system. The entries comprise user entries. The directory server has a password checking function capable of checking the password for a user entry, based on password-related data. The password checking function is responsive to a user entry having extra data associated thereto, and identifying an additional entry, for executing a distinct password checking based on the password related data defined in that additional entry.

This invention may also be defined as an apparatus or system, and/or as software code for implementing the method, or for use in the system, and/or as portions of such software code, in all their alternative embodiments to be described hereinafter.

Other alternative features and advantages of the invention will appear in the detailed description below and in the appended drawings, in which :

- Figure 1 is a general diagram of a computer system in which the invention is applicable;
- Figure 2 illustrates a typical LDAP exchange between a LDAP client and a LDAP server, and between the LDAP server and further servers;
- Figure 3 illustrates the general structure of a LDAP directory;
- Figure 4 shows a portion of a LDAP tree;
- Figure 5 represents attribute types and values of an entry;
- Figure 6 represents three types of LDAP roles according to the prior art;
- Figure 7 represents three types of LDAP classes of service according to the prior art;
- Figure 8 illustrates the structure of a class of service, according to the prior art;
- Figure 9 is a schema representing a structure for multiple password policies, in accordance with an embodiment of this invention;
- Figure 10 illustrates the generation of a special role-based attribute for multiple password policies, according to an embodiment of this invention;
- Figure 11 represents a portion of a directory tree in which the special role-based attribute for multiple password policies is generated, in accordance with an embodiment of this invention;
- Figure 12 is a flowchart for executing password policy checkings on a given user entry, according to an embodiment of this invention; and
- Figure 13 illustrates the subentry mechanism for scoping, defined in the ISO/IEC X.509 standard.

Additionally, the detailed description is supplemented with the following Exhibits:
- Exhibit E1 contains definitions of Password Policy attributes;
- Exhibit E2 contains definitions of Lockout Password Policy attributes;
- Exhibit E3 contains definitions of operational Password Policy attributes;
- Exhibit E4 contains example entry definitions for generating Password Policy attributes;
- Exhibit E5 contains definitions related to *PasswordPolicySubentry* attribute.

In the foregoing description, references to the Exhibits are made directly by the Exhibit or Exhibit section identifier: for example, E2.1 refers to section E2.1 in Exhibit E2. The Exhibits are placed apart for the purpose of clarifying the detailed description, and of enabling easier reference.

Now, making reference to software entities imposes certain conventions in notation. Particularly, an expression indicated between the quote signs " " may be used to design LDIF extracts and an expression in italics may be used for representing an attribute and an object class.

As they may be cited in this specification, Sun, Sun Microsystems and Sun One are trademarks of Sun Microsystems, Inc.

This invention may be implemented in a computer system, or in a network comprising computer systems. Figure 1 represents an example of the hardware of such computer systems. The hardware comprises :
- a processor CPU 11, e.g. an Ultra-Sparc;
- a program memory 12, e.g. an EPROM, a RAM, or Flash memory;
- a working memory 13, e.g. a RAM of any suitable technology;
- a mass memory 14, e.g. one or more hard disks;

- a display 15, e.g. a monitor;
- a user input device 15, e.g. a keyboard and/or a mouse;
- a network interface device 21 connected to a communication medium 20, which is in communication with other computers. Network interface device 21 may be of the type of Ethernet, or of the type of ATM. Medium 20 may be based on wire cables, fiber optics, or radio-communications, for example.

Data may be exchanged between the components of figure 1 through a bus system 10, represented as a single bus for simplification of the drawing. Bus systems may include a processor bus, e.g. PCI, connected via appropriate bridges to, e.g. an ISA or a SCSI bus.

The data exchanged are handled by a resource provider using a server to deliver data to user computers, or to store the data provided by the user computers. Browsers, e.g. Internet Explorer, are further provided on user computers, to enable users to make requests, to retrieve or store data. The resource provider makes it possible for user computers on a network to share data of any kind.

iPlanet E-commerce Solutions, now Sun One E-commerce Solutions, has developed a "net-enabling" platform called the Internet Service Deployment Plateform (ISDP). ISDP includes multiple, integrated layers of software that provide a full set of services supporting application development e.g. business-to-business exchanges, communications and entertainment vehicles, and retail Web sites.

Sun One™ Directory Server, provides a centralized directory service directory service for an intranet or an extranet. A directory service represents a collection of software, hardware, and processes that are able to deliver and store information. The directory service generally includes one or more directory client programs that can access the data stored in the directory, e.g. names, phone numbers or addresses.

The Sun One Directory Server is a general purpose directory that stores all information in a single, network-accessible repository. The Sun One Directory Server provides the standard protocol LDAP and an application programming interface (API) to access the information contained by the Sun One Directory Server.

LDAP is the Internet standard for directory lookups, just as the Simple Mail Transfer Protocol (SMTP) is the Internet Standard for delivering e-mail and the Hypertext Transfer Protocol (HTTP) is the Internet standard for delivering documents. Technically, LDAP is defined as on-the-wire bit protocol (similar to HTTP) that runs over Transmission Control Protocol/ Internet Protocol (TCP/IP). It specifies the interaction between clients and servers and determines how LDAP queries and responses are carried over the IP network.

A LDAP-compliant directory, such as the Sun One Directory Server, leverages a single, master directory that contains all users, groups and access information. The directory is hierarchical, not relational and is particularly fitted for reading while offering a high reliability and a high scalability.

Referring now to figure 2, LDAP defines a communication 1 between a server 17 and a client 18. LDAP also defines a communication 2 between LDAP server 17 and servers 17.1 to 17.n, which makes it possible for the server LDAP 17 to exchange its content (replication service) with servers 17.1 to 17.n or to access the directory of one of the servers 17.1 to 17.n (referral service) and vice versa.

The LDAP protocol is a message-oriented protocol. The client 18 constructs a LDAP message containing a request and sends the message to the server 17. The server 17 processes the request and sends a result, or results, back to the client 18 as a series of LDAP messages.

Such a client-server communication additionally lies on a specific architecture. LDAP creates a standard defining the way data are exchanged between the client computer and the directory server and defining the way data are modeled. More specifically, LDAP relies on four basic models:

- an information model;
- a naming model;
- a functional model; and
- a security model.

The LDAP information model defines the kind of data that can be stored in a directory.

LDAP directory is populated with entries. An entry corresponds to real-world objects, such as a person, a printer, or configuration parameters.

Figure 3 illustrates the general structure of a LDAP directory : the directory server 30 executes implemented functions based on the entries 31 stored in databases. The entries comprise configuration entries 310, user entries 311 and administrative entries 312. These entries further interact with the schema 32 described below. The configuration entries are stored under the subtree "cn=config". The user entries comprise data related to the users of the directory server. Administrative entries relate to user management and are generally implemented as LDAP subentries.

An entry contains a set of attributes associated with values. Each entry is uniquely identified by a distinguished name. The distinguished name may be stored in the *dn* attribute (distinguishedName).

LDAP entries are organized in a hierarchical tree structure, called the Directory Information Tree (DIT). Each node of the tree comprises an entry. Figure 4 illustrates an organization entry (22) with the attribute type of domain component *dc*, an organizational unit entry (24) with the attribute type of organizational unit *ou*, a server application entry (26) with the attribute type of common name *cn*, and a person entry (28) with the attribute type of user ID *uid*. The entries are connected by the directory. Each server has a particular entry called root directory specific entry (rootDSE) which contains the description of the tree and of its content.

A LDAP Data Interchange Format (LDIF) is an ASCII text file format used to describe directory entries and operations on those entries. It enables to create, modify, and delete Directory entries and to import and export data among LDAP directories. Figure 5 is a LDIF representation of an entry 404, showing the attribute types 400 and their values 402.

The information model is extensible, which means that new types of information can be added to a LDAP directory.

Descriptive information is stored in the attributes of the entry. Each attribute describes a specific type of information. Attributes may have constraints that limit the type and length of data placed in attribute values.

All entries require the *objectclass* attribute which lists the object classes to which an entry belongs. An entry can belong to one or more object classes and must satisfy all of them. The *objectclass* attribute defines which attributes are required and which attributes are allowed in the entry.

For example, in figure 5, the entry (404) represented in LDIF belongs to the object classes *top, person, organizationalPerson* and *inetOrgPerson.*

Each attribute has a corresponding syntax definition. The syntax definition describes the type of information provided by the attribute. The object classes, the required and allowed attributes, and the syntax definition of the attributes are listed in the directory schema.

The LDAP directory comprises a structure 32, represented in figure 3, that defines object classes and attributes. This structure, called the schema, sets the rules defining what information can be stored in the LDAP directory and how information is organized. The schema specifies the required and allowed attributes that are used to store information and their syntax definition. A schema checking function may be activated, thus causing the directory server to check new entries to verify whether :

- object classes and attributes attached to new entries are defined in the schema 32;
- the attributes required for an object class according to the schema 32, are contained in an entry attached to that object class,
- only attributes allowed by the object class according to the schema 32, are contained in an entry attached to that object class.

The LDAP naming model specifies that directory entries must be hierarchical and organized in an inverted tree structure. As mentioned above, each entry has a unique name called a distinguished name *dn*. The *dn* consists of a list of the names of all the parent entries in the directory back to the top of the directory hierarchy, the name of the entry being at the extreme left, e.g., "uid=Joe,ou=people,dc=france,dc=sun,dc=com", in figure 5. The root of the entry is at the extreme right of the *dn*. The name at the extreme left of the *dn*, "uid=Joe" in the example, is the relative distinguished name or *rdn*. Within a set of entries sharing the same parents, the *rdn* must be unique. This ensures that two entries in the directory tree cannot have the same *dn*.

The LDAP functional model comprises eight basic functional operations (indicated thereinafter between the quote signs " ") that a user from a client computer can perform on the directory data of a LDAP directory server :

- "bind" and "unbind" : begin and end the exchange of information between LDAP clients and the directory server;
- "add", "delete", and "modify" : apply on specific entries in the DIT,
- "compare" : applies on two entries to compare their content according to criteria,
- "search" : locates specific entries in the DIT,
- "modifyRDN" : applies to change the distinguished name *dn* of an entry.

In addition to the eight basic functional operations, the LDAP protocol defines a framework for adding new operations to the protocol via LDAP extended operations. Extended operations allow the protocol to be extended in an orderly manner to meet new marketplace needs as they emerge.

According to another aspect of LDAP directories, entry grouping mechanisms are provided to simplify the management of LDAP users. Roles constitute one of those grouping mechanisms. A role may have members, which are the entries said to possess the role. Role mechanisms enable the following operations:

- enumerating the members of a given role,
- determining whether a given entry possesses a particular role,
- enumerating all the roles possessed by a given entry,

It is further possible to assign a particular role to a given entry and to revoke a particular role from a given entry. Roles can also be associated with permissions, which allows a role-by-role access control management instead of a user-by-user access control management.

Every role is defined by its own definition entry. A role is uniquely identified by the distinguished name of its defining entry. Role definition entries are LDAP subentries and therefore inherit the subentry mechanism, defined in the ISO/IEC X.509 standard, for scoping.

Referring to figure 6, a role can be of "managed" type 601, "filtered" type 602 or "nested" type 603. Each type of role further has two specific object classes 61 that inherit from the *nsRoleDefinition* object class.

Roles can be used with a Class of Service (CoS) to provide role-based attributes. To illustrate the CoS concept, we can consider a directory containing thousands of entries that all share the common attribute *PostalCode*. Traditionally, to change the postal code represented by the common attribute *PostalCode*, an individual update of each entry is required, which constitutes a heavy job for administrators. With CoS, the generation of the attribute *PostalCode* is dynamic. The *PostalCode* attribute is stored in one place, and each entry points to that place to give a value to their postal code attribute. For the application, the attributes generated by CoS appear just like all other attributes, despite that they are not actually stored on the entries themselves. When coupled with roles, a CoS makes it possible to generate or update an attribute or role-based attribute for all the entries possessing the role.

Figure 8 illustrates the LDAP structure of a class of service. A class of service CoS is composed of the following entries in a LDAP directory:

- a CoS Definition Entry 941 that identifies the type of CoS. It is stored as a LDAP subentry below the branch it affects. The CoS definition entry more specifically identifies a CoS Template entry of the tree structure and target entries.
- a CoS template Entry 9810 that gives values to an attribute, identified by a *cosAttribute* in the CoS definition entry, the values being automatically generated in target entries 991.
- Target entries 991 providing the attribute and attribute values dynamically generated by the CoS definition entry and the template entry. Target entries share the same parent as the CoS definition entry.

Different types of CoS can be used depending on the way the value of dynamic attributes need to be generated.

There are three types of CoS:

- A pointer CoS comprising a pointer *cosTemplateDN* identifies the template entry using the template distinguished name only (link 3).
- An indirect CoS designating the template entry using the value of one of the target entry's attributes, identified by *cosIndirectSpecifier* attribute (link 8).
- A classic CoS identifying the template entry by both its distinguished name designated by *cosTemplateDN* attribute (link 3) and the value of one of the target entry's attributes designated by *cosSpecifier* attribute *(link 8)*.

The schema of figure 7 is a table representing the object classes 71 and the attributes 72 of a CoS definition entry 701 and of a CoS template entry 702 belonging to a classic CoS.

A CoS definition entry of a classic CoS belongs to the object classes *cosSuperDefinition* and *cosClassicDefinition* and comprises the attributes:

- *cosTemplateDN* identifying the distinguished name of the parent entry of the cos template entry,
- *cosSpecifier* identifying the value of one of the target entry's attributes, and
- *cosAttribute* identifying the attribute to generate in the target entry based on the values of the template entry.

A classic CoS is capable of generating attribute values for an entry based on the role possessed by the entry, which avoids generating the attribute values for each one of the entries of the role. Referring to figure 10, the structure for generating role based-attributes comprises a classic class of service having a CoS definition entry 940, CoS template entries 980 and Cos target entries 990, and role entries 970 interacting with the classic Class of Service.

A role-based attribute appears on an entry because the entry possesses a particular role associated with a template entry. The template entry generates the value of the attribute for all the entries that are members of that role. To update the role-based attribute on all the entries that are members of the role, one only needs to update the attribute in the template entry.

LDAP security model further provides LDAP controls for additional information to be supplied by users as part of a LDAP operation. One example of a LDAP control is a password policy control. Password policies are used to make it difficult for password cracking programs to break into the directory. For instance, a password policy control can be used to return information during a bind request concerning the password expiration state of the user's password.

A password policy is a set of rules that define how passwords are used in a given system. The password policy mechanism provided by the directory server allows to control password parameters such as how short a password must be and whether users can reuse passwords. When users attempt to bind to the directory, the directory compares the password with the value in the password attribute of the user's directory entry to make sure they match. The directory server also uses the rules defined by the password policy to ensure that the password is valid before allowing the user to bind to the directory.

LDAP password policies comprise control mechanisms, such as User-Defined Passwords, Password Change After Reset, Password Expiration, Expiration Warning, etc.

Password policies further comprise password policy attributes which set the parameters of passwords policies mechanisms. For instance, the password attribute *passwordMaxFailure* specifies the maximum of consecutive failed bind attempts after which a user account will be locked. Password policy attributes comprise the attributes that belong to the *passordPolicy* object class defined in Exhibit E1, such as *PasswordMustChange*, *PasswordChange*, *PasswordMinAge*, *PasswordExp*, etc.

Referring to figure 3, the schema 32 contains the definition of the *passwordPolicy* object class, as well as the definition of the password policy attributes. The *passwordPolicy* object class, contains a set of administrative password policy attributes.

Lockout password policy attributes are also defined in the schema. They prevent dictionary attacks against the password by counting the number of failed bind and locking the user's account. Exhibit E2 comprises the definition of lockout password policy attributes (*PasswordLockout*, *PasswordMaxFailure* and so on)

Operational password policy attributes are further defined in the schema 32. Exhibit E3 comprises the definition of these attributes (*PasswordHistory,passwordAllowChangeTime* and so on). A user entry can thus comprise these attributes without belonging to a particular object class. Operational password policy attributes are specific to an entry, non user-modifiable and usually computed based on password Policy attributes.

According to the prior art, the values of password policy attributes are defined for all the entries of the DIT. More specifically a password policy configuration entry 311 of the DIT assigns values to password policy attributes. General configuration entries are stored under the "cn=config" entry.

The values of the password policy attributes defined in this configuration entry apply to all the user entries of the DIT, according to the subentry mechanism for scoping, defined in the ISO/IEC X.509 standard. As a result a single password policy is applicable whatever may be the user entry, which limits password mechanism controls.

Particularly, it is not possible to define specific password policy attribute values for a user according to his role or according to specific user information. The existing password policy lacks granularity for administrators who may need to apply different password policies depending on the specificity of a user.

A solution of the prior art would be to directly define the password policy attributes and attribute values in each user entry. This would require to further assign the *passwordPolicy* object class to each user entry. This solution would be extremely complicated to implement and to administer. Indeed, it would require defining as many password policy attributes values as existing user entries, and each time adding the *passwordPolicy* object class to the user entry.

The invention addresses the above problems.

The invention proposes a directory server system capable of assigning a specific password policy to a user or a set of users, so as to adjust the password mechanism controls to the user profile.

More specifically, the invention proposes a password policy based on multiple password policy configuration entries, and no more based on a single password policy configuration entry. In the prior art, the single password policy configuration entry defined under "cn=config" applies to any user entry, independently of the attributes comprised by the user entry, as "cn=config" is located at the root of the tree structure.

Referring to figure 9, a directory tree structure according to the invention may comprise a set of password policy entries 91, each password policy entry defining a specific password policy. According to the invention, the directory server is capable of associating a given user entry E0 from user entries 991, with a particular password policy entry P1 from password policy entries 911, in order to apply the password policy defined by entry P1 to the user entry E0.

More specifically, user entries may be previously attached a special attribute identifying one of the policy password policy entries. As a result the directory server is operable for associating a given user entry E0 with a particular password policy P1 in response to the given entry E0 comprising the special attribute, and to the special attribute identifying the password policy entry P1 (link 2). Moreover, the directory server is also operable for executing password policy checkings based on the values of the password policy attributes defined in entry P1, in response to the password policy entry E0 being identified (link 4).

In a first embodiment of the invention, the special password policy attribute, hereinafter named *PasswordPolicySubentry,* is an operational one, and therefore is not associated with an object class in the schema. As a result, any user entry may be added this special attribute independently of the object classes to which the user entry belongs.

According to another embodiment of the invention, the value of *passwordPolicySubentry* attribute has a distinguished name (DN) syntax. This DN corresponds to the distinguished name of one of the password policy entries and thus identifies the password policy attribute values to be used for executing password policy checkings.

Exhibit E5.1 comprises an example of *PasswordPolicySubentry* attribute definition, in the schema of iPlanet (now "Sun One") Directory server.

An exemplary password policy entry is shown in Exhibit E4.1. A password policy entry, according to the invention, comprises :

- a distinguished name identifying the entry, e.g. "dn : pwp_1, <suffix>",
- the *passwordPolicy* object class ("object class : passwordPolicy "),
- a set of password policy attributes and attribute values, e.g. "PasswordMinAge : 0".

The set of password policy attribute and attribute values characterizes a password policy entry.

The *passwordPolicySubentry* attribute is attached to user entries in order to identify an associated password policy entry and thus the password policy attributes values to apply to each user.

Defining the password policy attribute values out of the user entry makes it easier for the administrator to add/ modify or delete these values.

Figure 12 is a flowchart representing the operations performed by the directory server for executing the password policy checkings on a given user entry E0.

In response to a bind request, or a modify request on a user's password attribute (*UserPassword*) at operation 101, the directory server gets the corresponding user entry E0 at operation 103.

At operation 105, the directory server checks whether user entry E0 has the *PasswordPolicySubentry* attribute.

If user entry E0 is attached to *PasswordPolicySubentry* attribute, the directory server gets the password policy entry P1, whose distinguished name is the value of *PasswordPoli-*

cySubentry attribute, at operation 107.

If entry P1 exists (test 109), at operation 113 the directory server gets the password policy attributes values defined in entry P1. At operation 121, the server executes the password policy checkings based on the password policy attribute values obtained at operation 113.

If entry E0 is not attached to *PasswordPolicySubentry* attribute (test 105) or if entry P1 does not exist in the DIT (test 109), at operation 111, the directory server gets a default password policy entry D0, defined in the configuration 311 under entry "cn= password policy, cn=config ". This entry comprises predefined values for password policy attributes that are applicable to all the user entries. Exhibit E4.2 comprises an example of a default password policy entry.

If the default password policy entry D0 is not present, the directory server gets "hard-coded" password policy attributes values at operation 117.

If the default password policy entry D0 is present, the directory server gets the password policy attributes values defined in this default entry, at operation 119.

At operation 121, the directory server then executes password policy checkings based on the password policy attribute values obtained at operation 117 or 119.

In addition, the invention defines a mechanism for attaching *PasswordPolicySubentry* attribute and a specific value for that attribute to a user entry or to a set of user entries. This mechanism depends on whether *PasswordPolicySubentry* is real or generated by a class of service.

In an embodiment of this invention, *PasswordPolicySubentry* attribute is generated by a classic class of service based on the roles possessed by the user entry.

According to the prior art, a classic Class of Service is capable of generating attribute values for an entry based on the role possessed by the entry, which avoids generating the attribute values for each one of the entries of the role.

According to an embodiment of the invention, with reference to figure 10, the *PasswordPolicySubentry* attribute is generated as a role-based attribute. More specifically, *cosAttribute* in the CoS definition entry designates *PasswordPolicySubentry* and CoS template entries give values to *PasswordPolicySubentry* attribute. Thus, instead of directly attaching this attribute to user entries, *PasswordPolicySubentry* attribute is generated based on the roles possessed by user entry E0.

As *PasswordPolicySubentry* attribute is operational, the template entries 980 that generate this attribute are not added a special object class.

According to the prior art, generated attributes or role based-attributes are obtained using the *nsrole* attribute as the value of *cosSpecifier* attribute in the CoS Definition entry.

With reference to figure 10, a role-based attribute is generated by the following operations of the prior art:

- computing *nsrole* for a given target entry E0,
- from the roles 970 identified by *nsrole*, and from the *cosTemplateDN* value determining the CoS Template entry T1,
- getting in this CoS template entry T1 the value of the attribute identified by *cosAttribute* in the CoS Definition entry 940, and generating this value in the target entry E0.

According to the prior art, the role entries 970, the CoS definition entry 940, the template entry identified by *cosTemplateDN* attribute should be located in the same level in the directory tree.

Figure 11 illustrates a portion of a tree structure representing an exemplary classic CoS generating *PasswordPolicySubentry* attribute and attribute value for a target entry, in accordance with the invention. Exhibit E4.3 contains the LDIF definition of the represented entries.

The user entry 9900 "cn=rob"is a target entry of the class of service defined by the CoS definition entry 9400.

The CoS attribute *PasswordPolicySubentry* ("cosAttribute =PasswordPolicySubentry ") is generated as indicated by "CosSpecifier : nsRole ".

The user entry 9900 possesses the role entry 9700, as is indicated by "nsRoleDN: cn=nsPwpExampleRole, <suffix>". Thus, this role will be a value of *nsrole* attribute, when computing *nsrole* for entry 9900, according to the operations described above.

The distinguished name of role 9700 "cn=nsPwpExampleRole,<suffix>" and the *cosTemplateDN* value "cn=nsPwpTmp" provide a distinguished name "cn=nsPwpExampleRole,<suffix>, cn=nsPwpTmp " that identifies the CoS template entry 9800.

The CoS attribute identifies the attribute *PasswordPolicySubentry* and the CoS template entry 9800 determines the value of this attribute "PasswordPolicySubentry:cn=pwp_1, <suffix>". *PasswordPolicySubentry* attribute and its value are then generated for the user entry 9900.

Consequently, user entry 9900 is attached *PasswordPolicySubentry* attribute with the value "cn=pwp_1, <suffix>".

The directory server then applies the flowchart of figure 12 for determining the password policy to apply to user entry 9900. For example, with reference to figure 11, the directory server may determine that the password policy entry 9100 identified by "cn=pwp_1, <suffix>" exists in the DIT, and therefore apply the value <attribute1>, <attribute2>, <attribute2> and so on, as password policy attribute values for executing password policy checkings.

According to the invention, the password policy entry P1 should be located at the same level as the role entries 970, the CoS definition entry 940 and the template entry, identified by *cosTemplateDN* attribute.

Defining *PasswordPolicySubentry* attribute as a role-based attribute makes it possible for the administrator to dynamically assign the same group of entries without actually modifying them, and to modify the desired password policy by changing a single entry (the CoS template entry).

In an alternative embodiment according to the invention, *passwordPolicySubentry* attribute is real. The administrator defines directly in the user entry the *passwordPolicySubentry* attribute and attribute value. An example of a user entry comprising a real *passwordPolicySubentry* attribute is shown in Exhibit E4.4.

According to this embodiment, a user entry associated with a password policy entry should be located in the scope of that entry, which is defined as the subtree of the parent entry of the password policy entry. Figure 13 illustrates the subentry mechanism for scoping, defined in the ISO/IEC X.509 standard. User entries E01 and E02 are in the scope S1 of the password policy entry P1, and therefore can be associated with this entry. Inversely, user entry E11 is out of the scope of password policy entry P1 and therefore cannot be associated with entry P1.

Consequently, with reference to the flowchart of figure 12, operation 109 further comprises determining whether the user entry E0 is in the scope of the password policy entry P1, and if not, getting the default password policy entry at operation 111.

According to this embodiment, the administrator has to modify every single entry to add *passwordPolicySubentry* attribute and attribute value.

Additionally, the *passwordPolicySubentry* attribute must be controlled by an Access Control Instruction (ACI) in order to prevent an unauthorized user from modifying this attribute. An example of ACI is shown in Exhibit E5.2.

In the prior art, the password policy configuration was stored under "cn=config". As a result, it was not replicated and had to be the same on all the replicas. According to the invention, there is no more one global configuration for all the entries. The password policy may be entry-based or role-based and thus may be replicated. As for the default password policy entry that may be stored under "cn= password policy, cn=config", it should be the same on all the replicas.

According to another embodiment of the invention, the password policy configuration data from "cn=config" of the prior art may be migrated in the default password policy entry.

This invention also encompasses software code, especially when made available on any appropriate computer-readable medium. The expression "computer-readable medium" includes a storage medium such as magnetic or optic, as well as a transmission medium such as a digital or analog signal. Such software code may include data and/or metadata.

This invention further encompasses the software code to be added to existing directory server functionalities to perform anyone of the various new functionalities, as described above, which may be used independently of each other.

On another hand, a number of features have been positively described, using absolute language, to help understanding the LDAP example. Each such feature should be considered as exemplary only, and is not intended to restrict the scope of this invention in any way.

Exhibit E1- Password Policy attributes

Password policy attributes belongs to passwordPolicy object class. Their values indicates the parameters for determining when and how a user can/must change his password. The following sections E1.1 to E1.11 describe the main password policy attributes.

E1.2 - *passwordMaxAge*

This attribute holds the number of seconds after which a modified password will expire. If this attribute is not present, or if the value is 0 the password does not expire.

E1.3 - *passwordExp*

ON means that the *passwordExpirationTime* must be updated after a successful password modification (i.e, password expiration is set).

E1.4 - *passwordMinLength*

This attribute holds the minimum number of characters that must be used in a password, if syntax checking is enabled. If this attribute is not present, no minimum password length will be enforced.

E1.5 -*passwordInHistory*

This attribute specifies the maximum number of used passwords stored in the *passwordHistory* attribute. If this attribute is not present, or if the value is 0, used passwords are not stored in the *passwordHistory* attribute and thus may be reused.

E1.6 - *passwordChange*

This attribute indicates whether users can change their own passwords. If this attribute is not present, a value of ON is assumed.

E1.7 - *passwordWarning*

This attribute specifies the maximum number of seconds before a password is due to expire that expiration warning messages will be returned to an authenticating user. If this attribute is not present, or if the value is 0 no warnings will be sent.

E1.8 - *passwordCheckSyntax*

This attribute indicates how the password syntax will be checked while being modified or added. If this attribute is not present, or if the value is OFF, syntax checking will not be enforced. A value of OFF indicates that the server will check the syntax. In fact, only the *passwordMinLength* is checked.

E1.9 - *passwordMustChange*

This attribute specifies with a value of ON that users must change their passwords when they first bind to the directory after a password is set or reset by the administrator. If this attribute is not present, or if the value is OFF, users are not required to change their password upon binding after the administrator sets or resets the password.

E1.10 - *passwordStorageScheme*

This attribute holds the schema tag name used to hash the *userPassword* attribute.

E1.11 - *passwordMinAge*

This attribute holds the number of seconds that must elapse between modifications to the password. If this attribute is not present, 0 seconds is assumed.

Exhibit E2- Lockout policy attributes

2.1- *passwordLockout*

This attribute indicates, when its value is "ON", that users will be locked out of the directory after a specified number of consecutive failed bind attempts. The maximum number of consecutive failed bind attempts is specified in *passwordMaxFailure*. If this attribute is not present, or if the value is "OFF", the account will not be locked when the number of failed bind attempts has been reached.

2.2- *passwordMaxFailure*

This attribute specifies the number of consecutive failed bind attempts after which a users account will be locked. If this attribute is not present, or if the value is 0, the account will not be locked due to failed bind attempts and the value of *passwordLockoutDuration* will be ignored.

2.3- *passwordResetFailureCount*

This attribute holds the number of seconds after which the password failures are purged from the failure counter (=*passwordRetryCount*), even though no successful authentication occurred. If this attribute is not present, or if its value is 0, the failure counter is only reset by a successful authentication.

2.4- *passwordLockoutDuration*

This attribute holds the number of seconds that an account will remain locked due to too many failed bind attempts. If this attribute is not present, or if the value is 0 the account will be locked until reset by an administrator.

2.5- *passwordUnlock*

If passwordMaxFailure has been reached and the value of this attribute is OFF, it means that the account is locked until the administrator resets it. If *passwordMaxFailure* has been reached and the value is ON, the account is locked for *passwordLockoutDuration* seconds. If *accountUnlockTime* is 0 and the value of this attribute is OFF, the account is locked until the administrator resets it.

Exhibit E3- Operational Password Policy attributes

E3.1- *passwordHistory*

This attribute holds a history of previously used passwords.

E3.2- *passwordAllowChangeTime*

This attribute is used to specify the exact time after which the user is allowed to change his password.

E3.3- *passwordExpirationTime*

This attribute is used to specify the exact time after which the user's password expires.

E3.4- *passwordExpWarned*

This attribute contains the time when the password expiration warning was first sent to the client. The password will expire in the *passwordWarning* time.

E3.5-*passwordRetryCount*

This attribute is used to count the number of consecutive failed attempts at entering the correct password.

E3.6-*retryCountResetTime*

This attribute specifies the exact time after which the *passwordRetryCount* is reset.

E3.7- *accountUnlockTime*

This attribute refers to the exact time after which the entry can be used for authentication.

Exhibit E4- entry definitions according to the invention

E4.1 - Password Policy entry

```
dn: cn=pwp_1,<suffix>
objectclass: top
objectclass: passwordPolicy
objectclass: LDAPsubentry
passwordMinAge: 0
passwordMaxAge: 8640000
<attributes of passwordPolicy objectclass>
```

E4.2- Default password policy entry

```
dn: cn=Password Policy,cn=config
objectclass: top
cn : Password Policy
objectclass: passwordPolicy
passwordMinAge: 0
passwordMaxAge: 8640000
<attributes of passwordPolicy objectclass>
```

E4.3 - passwordPolicySubentry is virtual: Roles/COS entries

E4.3.1 - Role entry

```
dn: cn=nsPwpExampleRole,<suffix>
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsSimpleRoleDefinition
objectclass: nsManagedRoleDefinition
cn: nsPwpExampleRole
```

E4.3.2 - CoS Template entry indicated by the specifier

```
dn: cn=nsPwpTmp,<suffix>
objectclass: top
objectclass: nsContainer
```

E4.3.3 - CoS Template entry associated with role cn=nsPwpExampleRole

```
dn: cn=\"cn=nsPwpExampleRole,<suffix>\",cn=nsPwpTmp,<suffix>
objectclass: top
objectclass: extensibleObject
objectclass: costemplate
objectclass: ldapsubentry
```

```
cosPriority: 1
passwordPolicySubentry: cn=pwp_1, <suffix>
```

E4.3.4 - CoS definition entry

```
dn: cn=nsPwp_cos,<suffix>
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
cosTemplateDn: cn=nsPwpTmp,<suffix>
cosSpecifier: nsRole
cosAttribute: passwordPolicySubentry
```

E4.4 - passwordPolicySubentry attribute is real

```
dn: cn=users,<suffix>
objectclass: ...
passwordPolicySubentry : cn=pwp_1, <suffix>
```

Exhibit E5 - PasswordPolicySubentry attribute

E5.1- Definition in the schema

```
attributeTypes : (passwordPolicySubentry-oid NAME 'passwordPolicySuben-
try' DESC 'iPlanet defined password policy attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.12 X-ORIGIN 'iPlanet Directory Server' USAGE
directoryOperation)
```

E5.2-Access control

```
targetattr != \"passwordPolicySubentry\")(version 3.0; acl \""
"Allow self entry modification except for passwordPolicySubentry "
"allow (write)userdn =\"ldap:///self\";)";
```

Claims

1. A method of implementing a password checking function based on password-related data in a directory server system, in which the directory server system has a directory server interacting with entries organized in a tree structure, and the entries comprising user entries,
the method comprising the steps of :
a) creating an additional entry, having attached password-related data,
b) attaching extra data to a given user entry, the extra data designating the additional entry, and
c) upon a call of the password checking function for the given entry, executing the password policy checking function for the given user entry based on the password-related data in the additional entry designated by the extra data of the given user entry.

2 . The method of claim 1, wherein the additional entry of step a) has a scope in the tree structure, and step c) is performed subject to the given user entry belonging to the scope of the additional entry.

3. The method of claim 2, wherein the scope of the additional entry comprises the subtree of the parent entry of that additional entry.

4. The method as claimed in any of claims 1 through 3, wherein the additional entry of step a) has at least one attribute, whose value contains password-related data.

5. The method of claim 4, wherein the value of said at least one attribute contains password policy data.

6. The method as claimed in any of claims 1 through 5, wherein step a) comprises attaching an object class data to an additional entry, said object class data identifying a predefined object class associated with said password-related data.

7. The method as claimed in any of claims 1 through 6, wherein step b) further comprises checking whether the tree structure comprises the additional entry, as identified by the

extra data.

8. The method of claim 2, wherein:

- step a) comprises creating a plurality of additional entries, each having attached password-related data, the additional entries having different scopes in the tree structure, and

- step c) is performed subject to the given user entry belonging to the scope of one of the additional entries.

9. The method as claimed in claim 8, further comprising the step of :
d) upon step c) being not performed or failing for a user entry, executing the password policy checking function on that user entry, using predefined password related data.

10. The method of claim 1, wherein the extra data of step b) designates the location of said additional entry.

11. The method as claimed in any of claim 1 through 10, wherein step b) comprises directly adding the extra data to the given user entry.

12. The method of claim 11, wherein step b) comprises adding the extra data to the given user entry in the form of an extra data attribute whose value designates the location of the additional entry.

13. The method of claim 12, wherein the extra data attribute is an operational attribute.

14. The method as claimed in any of claim 1, in which the directory server has a class of service mechanism, wherein step b) comprises adding the extra data in the form of a class of service being applicable to one or more user entries.

15. The method of claim 14, in which a class of service has a scope and said one or more user entries are located in the scope of the class of service.

16. The method of claim 15, wherein the scope of a class of service comprises the entries

located in the subtree of the parent entry of an entry defining the class of service.

17. The method as claimed in any of claims 14 through 16, in which the directory server has a role mechanism, wherein step b) comprises adding the extra data to a class of service being applicable to one or more user entries having the same role.

18. The method as claimed in any of claims 14 through 17, wherein step a) comprises adding said additional entry in the same level of the tree as the entry defining the class of service.

19. A directory server capable of interacting with entries organized in a tree structure in a directory server system, said entries comprising user entries,
the directory server having a password checking function capable of checking the password for a user entry, based on password-related data,
said password checking function being responsive to a user entry having extra data associated thereto, and identifying an additional entry, for executing a distinct password checking based on the password related data defined in that additional entry.

20 . The directory server of claim 19, wherein the additional entry has a scope in the tree structure, and said distinct password checking is performed subject to the given user entry belonging to the scope of the additional entry.

21. The directory server of claim 20, wherein the scope of the additional entry comprises the subtree of the parent entry of that additional entry.

22. The directory server as claimed in any of claims 19 through 21, wherein the additional entry has at least one attribute, whose value contains password-related data.

23. The directory server of claim 22, wherein the value of said at least one attribute contains password policy data.

24. The directory server as claimed in any of claims 19 through 23, wherein step a) the extra data comprises object class data attached to the additional entry, said object class

data identifying a predefined object class associated with said password-related data.

25. The directory server as claimed in any of claims 19 through 24, further comprising a mechanism for checking whether the tree structure comprises the additional entry, as identified by the extra data.

26. The directory server of claim 20, wherein the entries comprise a plurality of additional entries, each having attached password-related data, the additional entries having different scopes in the tree structure, and said password checking function is responsive to a user entry belonging to the scope of one of the additional entries, for executing a distinct password checking based on the password related data defined in that one of the additional entries.

27. The directory server as claimed in claim 26, wherein said password checking function is adapted to execute the password checking function on a user entry, using predefined password related data, when that user entry has no extra data associated thereto.

28. The directory server of claim 19, wherein the extra data designates the location of the associated additional entry.

29. The directory server as claimed in any of claim 19 through 28, comprising extra data being directly added to the user entry.

30. The directory server of claim 29, wherein the extra data are added to the given user entry in the form of an extra data attribute, whose value designates the location of the additional entry.

31. The directory server of claim 30, wherein the extra data attribute is an operational attribute.

32. The directory server as claimed in any of claim 19 through 31, in which the directory server has a class of service mechanism, comprising the extra data being added in the form of a class of service being applicable to one or more user entries.

33. The directory server of claim 32, in which a class of service has a scope, wherein said one or more user entries are located in the scope of the class of service.

34. The directory server of claim 33, wherein the scope of a class of service comprises the entries located in the subtree of the parent entry of an entry defining the class of service.

35. The directory server as claimed in any of claims 32 through 34, in which the directory server has a role mechanism, comprising extra data being added in the form of a class of service being applicable to one or more user entries having the same role.

36. The directory server as claimed in any of claims 32 through 35, comprising said additional entry being added in the same level of the tree as the entry defining the class of service.

37. The software code for performing the steps of any of claims 1 through 18.

38. The software code for a directory server as claimed in any of claims 19 through 36.

39. A plug-in software code for a directory server, comprising additional code for a password checking, such that the password checking function responds to a user entry having extra data associated thereto, and identifying an additional entry, for executing a distinct password checking based on the password related data defined in that additional entry

40. The combination of the software code of claim 39 with the password checking function.

CABINET NETTER

[illegible]

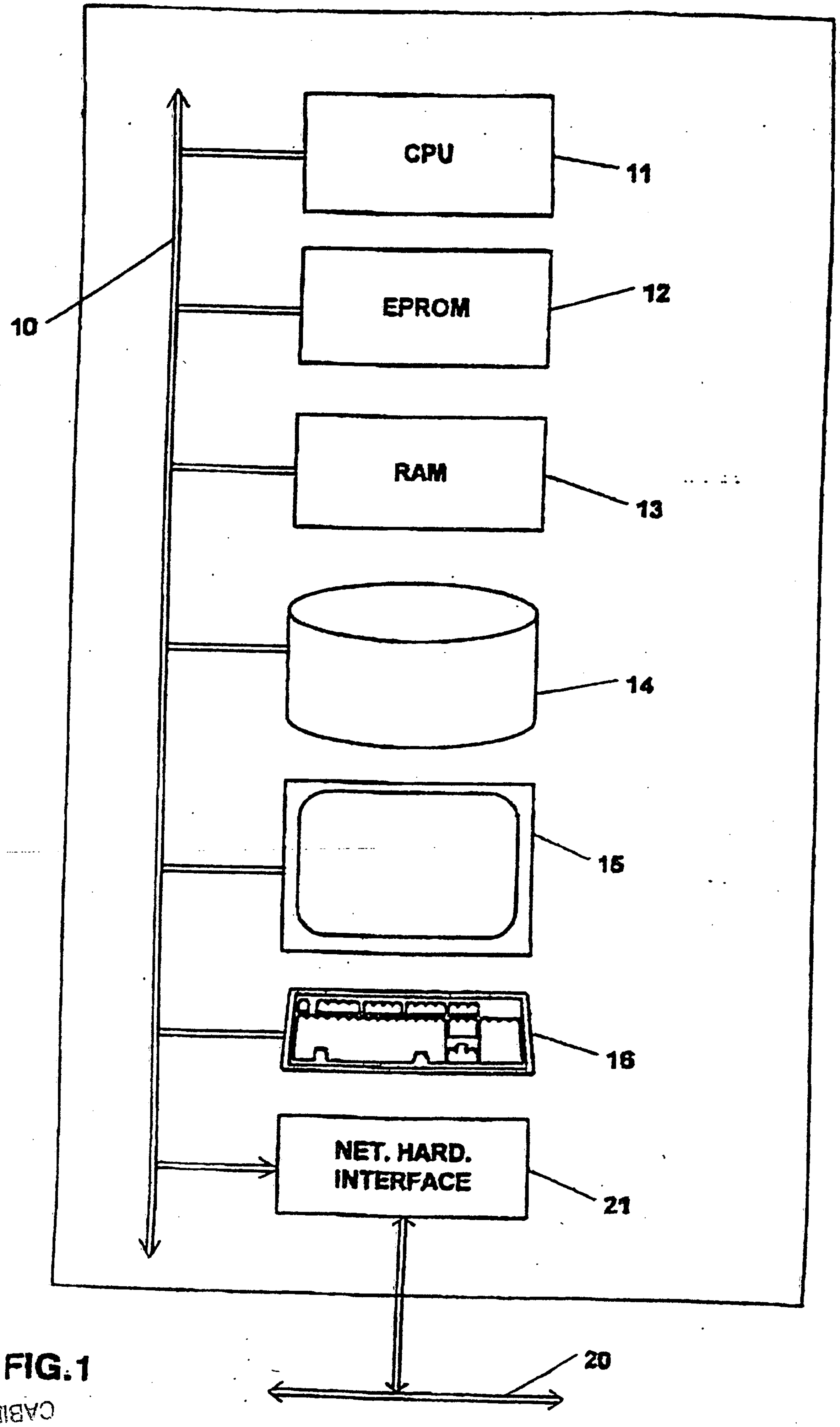


FIG.1

PRIOR ART

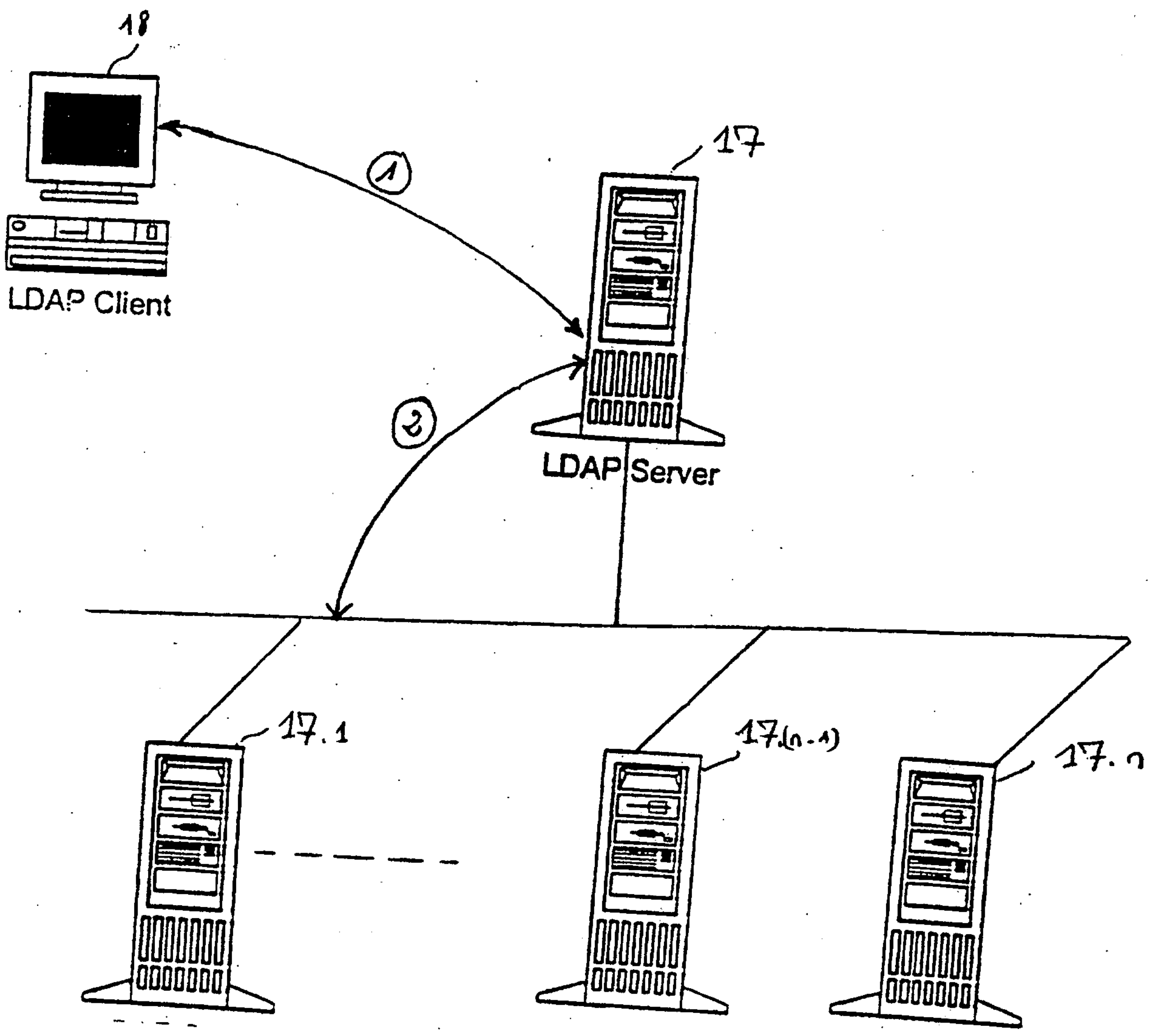


FIG.2.

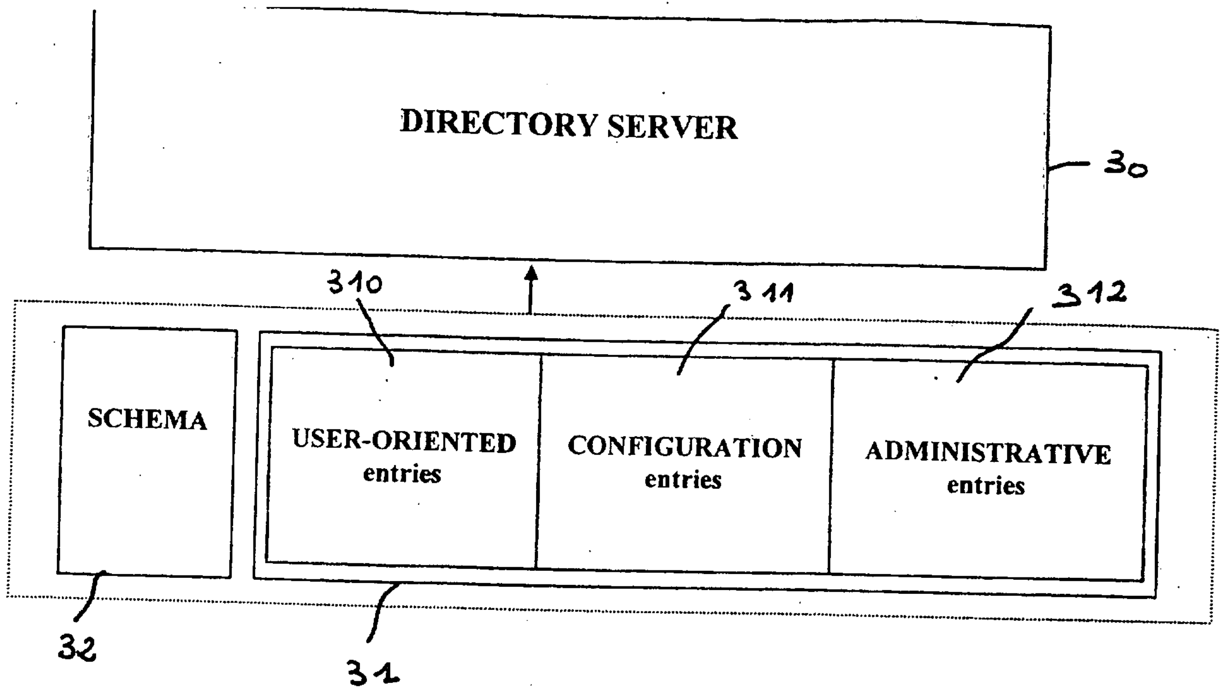


FIG 3

CABINET NETTER

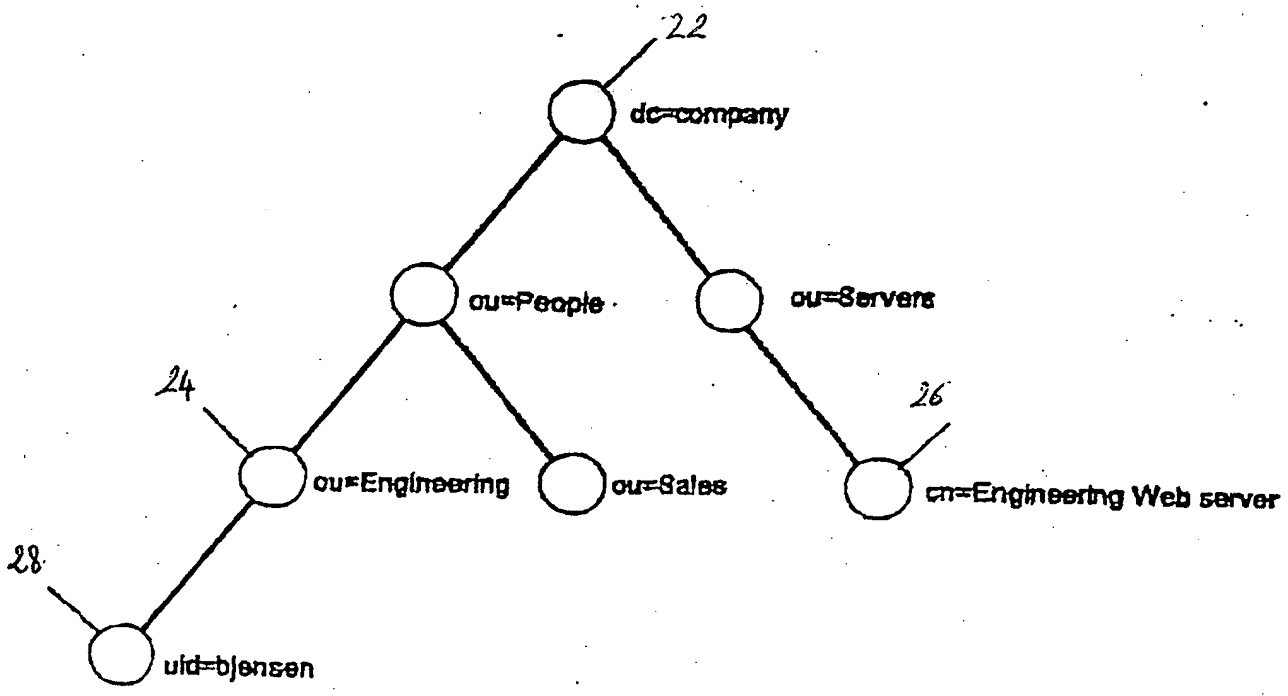


FIG 4

(PRIOR ART)

CABINET NETTER

Entry 404

Attribute type 400	Attribute value 402
dn :	uid=Joe, ou=people, dc=france, dc=sun, dc=com
objectClass :	top
objectClass :	person
objectClass :	organizationalPerson
objectClass :	inetOrgPerson
cn :	Joe
sn :	Rayan
uid :	joerayan
mail :	joerayan@sun.com
phoneNumber :	778

Fig 5

(PRIOR ART)

61 62

Role Type	Object Classes	Attributes
Managed Role	nsSimpleRoleDefinition nsManagedRoleDefinition	description (optional)
Filtered Role	nsComplexRoleDefinition nsFilteredRoleDefinition	NsRoleDN description (optional)
Nested Role	nsComplexRoleDefinition nsNestedRoleDefinition	NsRoleDN description (optional)

602
603
604

Figure 6

(PRIOR ART)

CoS entries	Object Classes	Attributes
701 **CoS definition entry**	cosSuperDefinition cosClassicDefinition	cosTemplateDN cosSpecifier cosAttribute
702 **CoS template entry**	cosTemplate	<cosAttribute> cosPriority

FIG 7
(PRIOR ART)

CABINET NETTER

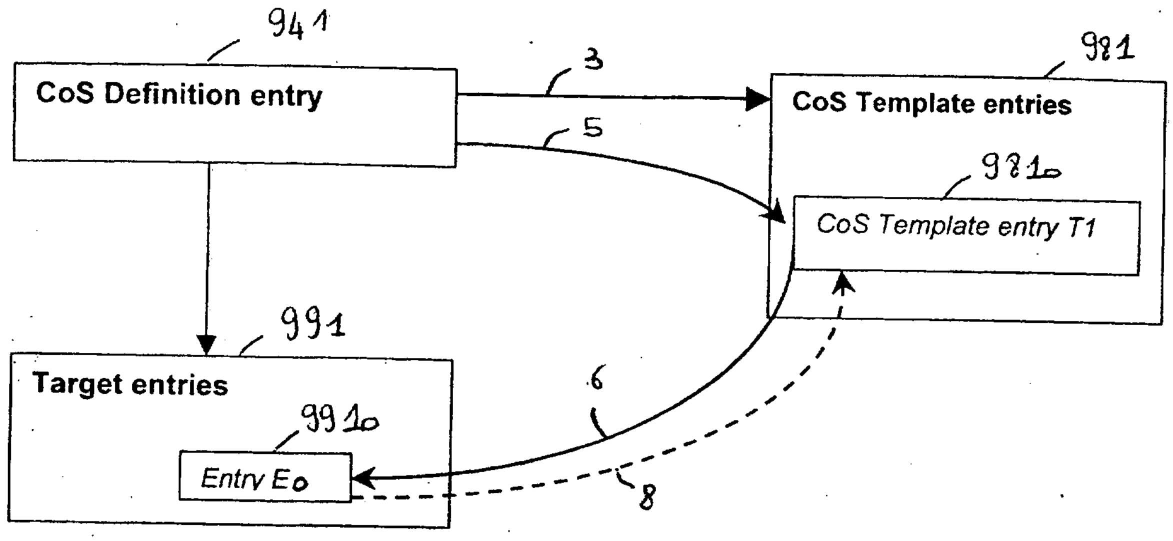


FIG 8

CABINET NETTER

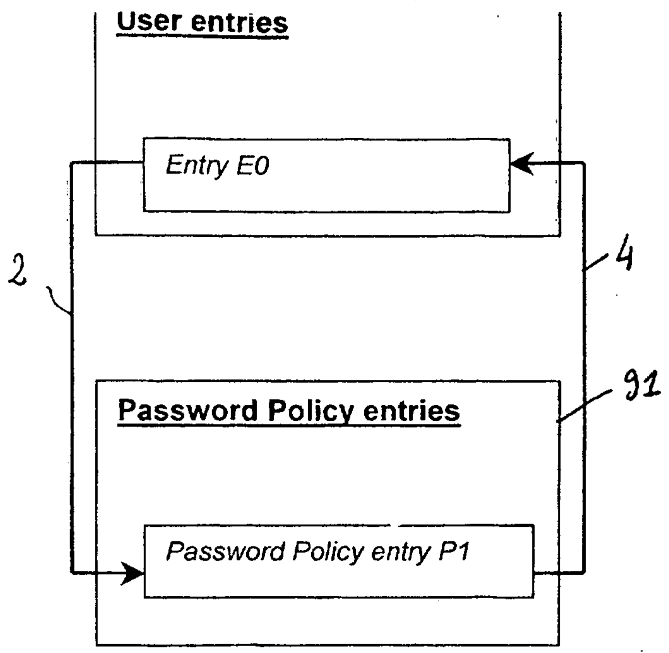
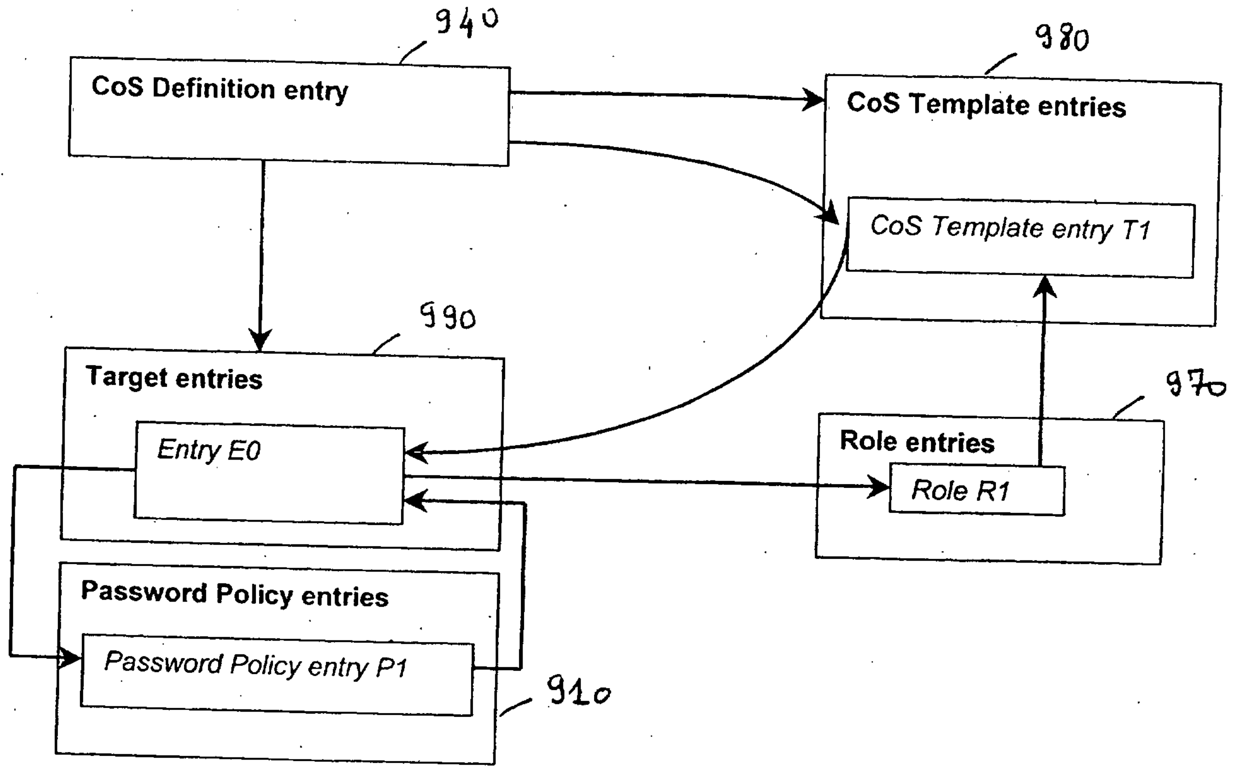

User entries
Entry E0
2
4
91
Password Policy entries
Password Policy entry P1

FIG 9



940
CoS Definition entry
980
CoS Template entries
CoS Template entry T1
990
Target entries
Entry E0
970
Role entries
Role R1
Password Policy entries
Password Policy entry P1
910

FIG. 10

CABINET NETTER

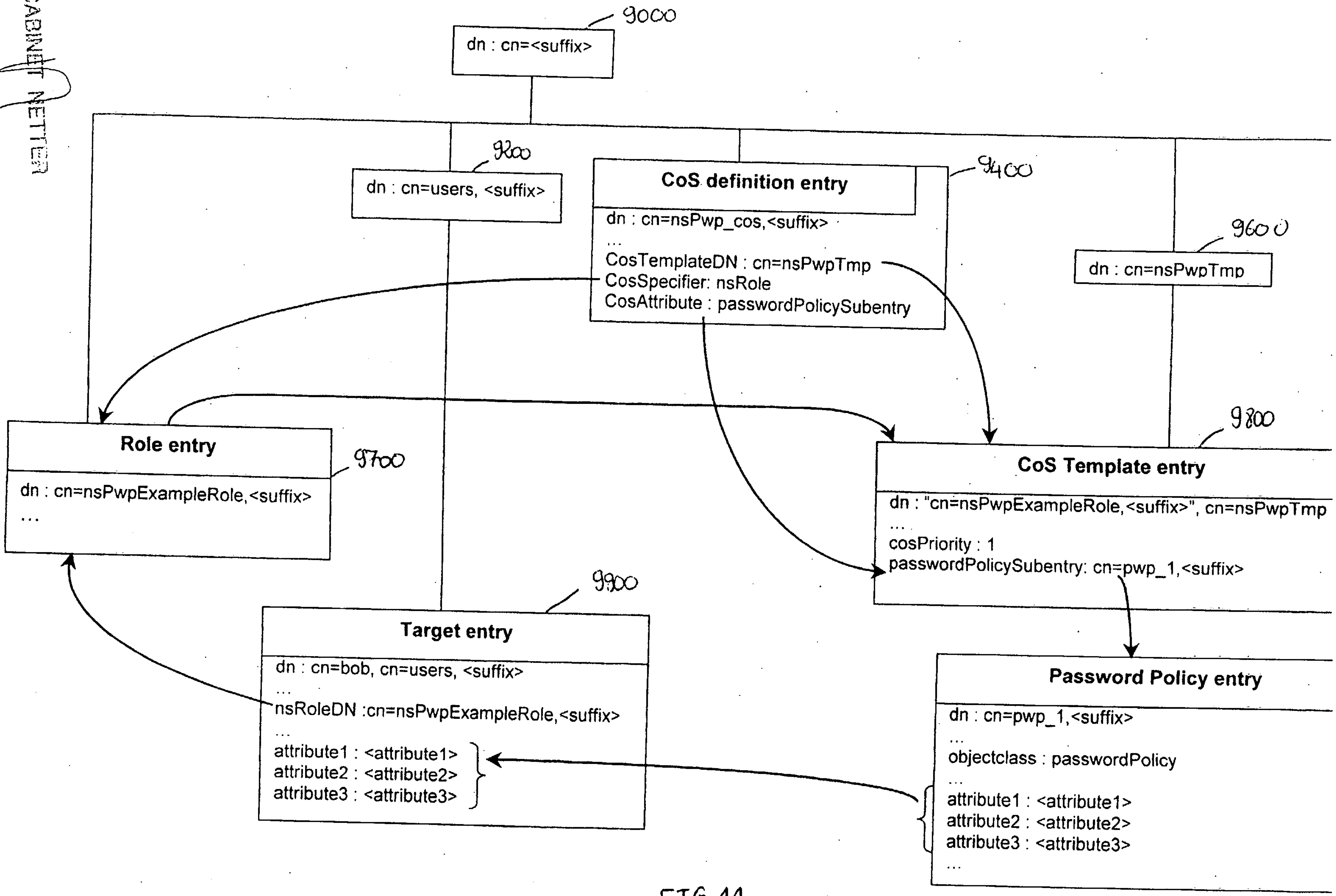

9000
dn : cn=<suffix>
9200
dn : cn=users, <suffix>
9400
CoS definition entry
dn : cn=nsPwp_cos,<suffix>
...
CosTemplateDN : cn=nsPwpTmp
CosSpecifier: nsRole
CosAttribute : passwordPolicySubentry
9600
dn : cn=nsPwpTmp
9700
Role entry
dn : cn=nsPwpExampleRole,<suffix>
...
9800
CoS Template entry
dn : "cn=nsPwpExampleRole,<suffix>", cn=nsPwpTmp
...
cosPriority : 1
passwordPolicySubentry: cn=pwp_1,<suffix>
9900
Target entry
dn : cn=bob, cn=users, <suffix>
...
nsRoleDN :cn=nsPwpExampleRole,<suffix>
...
attribute1 : <attribute1>
attribute2 : <attribute2>
attribute3 : <attribute3>
Password Policy entry
dn : cn=pwp_1,<suffix>
...
objectclass : passwordPolicy
...
attribute1 : <attribute1>
attribute2 : <attribute2>
attribute3 : <attribute3>
...

FIG 11

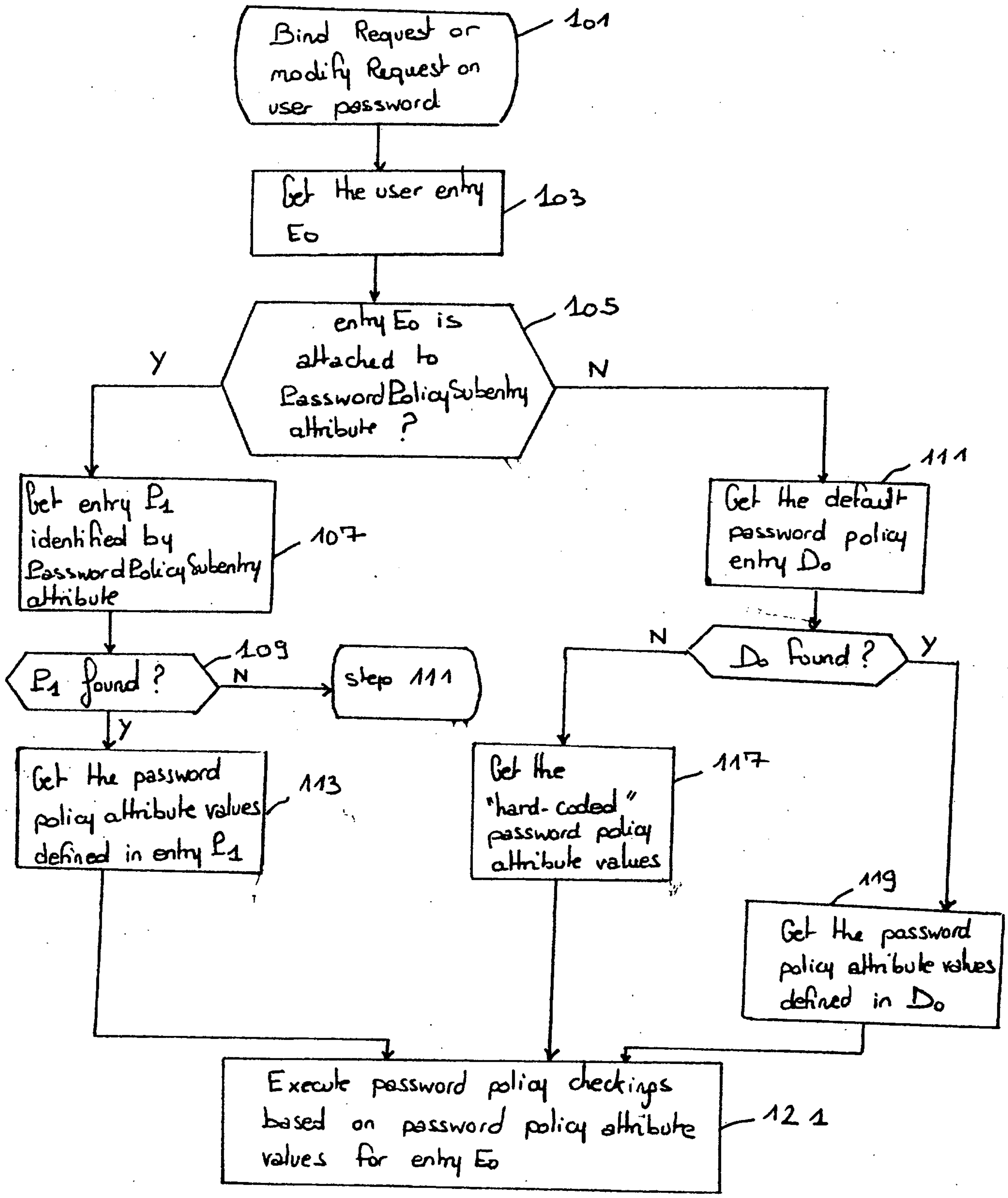

Bind Request or modify Request on user password
101
Get the user entry E₀
103
entry E₀ is attached to PasswordPolicySubentry attribute ?
105
Y
N
Get entry P₁ identified by PasswordPolicySubentry attribute
107
Get the default password policy entry D₀
111
P₁ found ?
109
N
Y
step 111
D₀ found ?
N
Y
Get the password policy attribute values defined in entry P₁
113
Get the "hard-coded" password policy attribute values
117
Get the password policy attribute values defined in D₀
119
Execute password policy checkings based on password policy attribute values for entry E₀
121

Figure 12

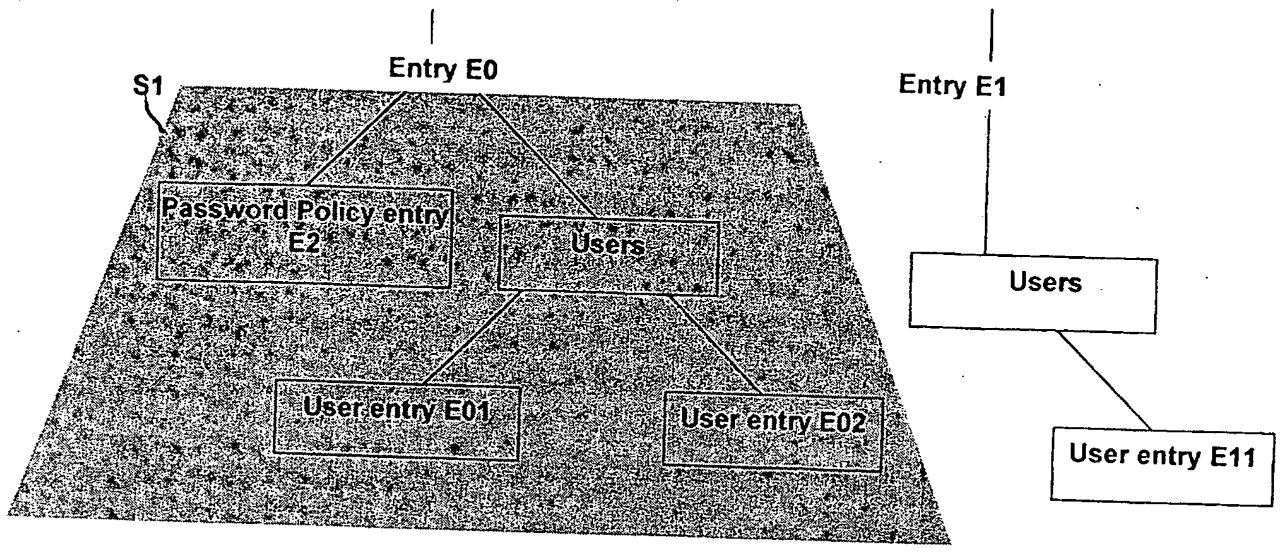


FIG 13

CABINET NETTER